Cryptography in Quadratic Function Fields

R. Scheidler*
Department of Mathematical Sciences
University of Delaware, Newark DE 19716, USA
scheidle@math.udel.edu

October 13, 1999
CORR #99-53

Abstract

We describe several cryptographic schemes in quadratic function fields of odd characteristic. In both the real and the imaginary representation of such a field, we present a Diffie-Hellman-like key exchange protocol as well as a public-key cryptosystem and a signature scheme of ElGamal type. Several of these schemes are improvements of systems previously found in the literature, while others are new. All systems are based on an appropriate discrete logarithm problem. In the imaginary setting, this is the discrete logarithm problem in the ideal class group of the field, or equivalently, in the Jacobian of the curve defining the function field. In the real case, the problem in question is the task of computing distances in the set of reduced principal ideals, which is a monoid under a suitable operation. Currently, the best general algorithms for solving both discrete logarithm problems are exponential (subexponential only in fields of high genus), resulting in a possibly higher level of security than that of conventional discrete logarithm based schemes.

1 Introduction

Since the introduction of the well-known Diffie-Hellman key exchange protocol [17], many cryptographic schemes based on discrete logarithms in a variety of groups and even semigroups have been proposed. Among them, the signature scheme due to ElGamal [18], now

*Research supported by NSF grant DMS-9631647

the basis of the U.S. Digital Signature Standard [30], is notable. ElGamal also presented a discrete logarithm based public-key cryptosystem in [18].

Any finite group or semi-group G with an efficient representation of elements, a fast operation and a sufficiently difficult discrete logarithm problem (DLP) lends itself to the use of discrete logarithm based cryptography. Diffie and Hellman as well as ElGamal used $G = \mathbb{F}_p^*$, the multiplicative group of a finite prime field. Buchmann et al. based a key exchange protocol on discrete logarithms in the ideal class group of an imaginary quadratic number field [11, 10]. The first example of a non-group underlying a discrete logarithm based system was the set $\mathcal{R}$ of reduced principal ideals of a real quadratic number field, which admits a structure first explored by Shanks [37] and termed *infrastructure* by him. A key exchange protocol using elements of $\mathcal{R}$ as keys was introduced in [12] and implemented in [35]. A signature scheme using the same set was briefly mentioned in [9]. These ideas were subsequently adapted to real quadratic function fields over finite fields, where the set of reduced principal ideals exhibits an analogous infrastructure. The first discrete logarithm based system in real quadratic function fields of odd characteristic was the key exchange protocol of [36], followed by a signature scheme in [34], and both a key exchange and a signature algorithm for even characteristic in [29]. Other schemes for real quadratic function fields and number fields, such as an oblivious transfer protocol, were discussed in [8].

The function field schemes are faster, simpler, and easier to implement than the corresponding number field systems. They use finite field arithmetic, thereby eliminating the need for rational approximations. The size of their underlying structure is determined by two parameters (the size of the field of constants and the genus of the function field) as opposed to only one parameter for number fields (their discriminant), so these two quantities can be chosen to optimize use of computer hardware. Finally, their underlying DLP can currently only be solved in exponential time, provided the genus of the field is not too large, whereas for the corresponding problem in number fields, a subexponential algorithm is known (provided the Extended Riemann Hypothesis is true). Thus, these fields seem to represent a promising setting for cryptography.

This paper considers a variety of discrete logarithm based cryptographic schemes in quadratic function fields of odd characteristic. We present Diffie-Hellman-like key exchange protocols as well as ElGamal-like signature schemes and public key systems in both the real and the imaginary model of such a field. Implementations of hyperelliptic systems (see [26]) using the ideal class group of an imaginary quadratic function field are discussed. For the real setting of a quadratic function field, we simplify the key exchange algorithm of [36]. We also improve on the signature schemes of [34] and [29] by shortening the signatures. Finally, we provide a more rigorous and meaningful complexity analysis of these systems than [36, 34, 29] and investigate their security and its relationship to the relevant discrete logarithm problems.

The fields under consideration are function fields $K = k(C)$ of an elliptic or hyperelliptic curve C of genus g over a finite field k of odd characteristic. If we write C in the form $y^2 = D(t)$ where $D(t)$ is a polynomial with coefficients in k, then $\deg(D(t)) = 2g + 1$ (*imaginary case*) or 2g+2 (*real case*). Here, we assume that the leading coefficient $\operatorname{sgn}(D)$ of $D(t)$ is a square in k^*; otherwise, just adjoin a square root of $\operatorname{sgn}(D)$ to k, obtaining a constant field isomorphic to $\mathbb{F}_{q^2}$, and consider K over the field of constants $\mathbb{F}_{q^2}$. It is a simple matter to convert an imaginary representation to a real one over the same field $k(t)$ of rational functions: for example, by replacing $D(t)$ by $D(t^{-1} + b)t^{2g+2}$ where b is chosen so that the leading coefficient $D(b)$ of the new polynomial is a square in k^*. The reverse is only possible if $D(t)$ has a root $a \in k$, corresponding to a ramified rational prime divisor of $K/k(t)$; we then replace $D(t)$ by $D(t^{-1} + a)t^{2g+2}$ which has degree $2g + 1$ and leading coefficient $D'(a)$ (here, $D'(t)$ is the formal derivative of $D(t)$ with respect to t). Hence the real setting is more general. A preliminary investigation [32] suggests that the arithmetic underlying our cryptographic schemes has approximately the same complexity and shows roughly equal performance in both the real and the imaginary model. A choice of setting for conducting cryptography (real versus imaginary) would depend on the performance issue as well as the question of how difficult the DLP is in either model. Currently, the best known general algorithms for solving this problem are exponential in both cases, and subexponential if the genus of the field is sufficiently large.

In the next section, we summarize the necessary basics about quadratic function fields. All the required algorithms and their complexity are stated in section 3. We present our cryptographic schemes in section 4 and analyze their security in section 5.

2 Quadratic Function Fields

For an introduction to function fields, we refer the reader to [41]. Quadratic function fields are discussed in considerable detail in [6]. Let $k = \mathbb{F}_q$ be a finite field of odd characteristic with q elements. A *quadratic function field* is a quadratic extension K of the rational function field $k(t)$ over k in the variable t. More specifically, $K = k(t, \rho)$ where t is transcendental over k and $\rho^2 = D$ with $D = D(t)$ a polynomial in t with coefficients in k which we may assume to be squarefree. K is (a) *real* (quadratic function field) if the degree $\deg(D)$ of D is even and the leading coefficient $\operatorname{sgn}(D)$ of D is a square in k. K is (an) *imaginary* (quadratic function field) otherwise; that is, K is imaginary if $\deg(D)$ is odd or $\deg(D)$ is even and $\operatorname{sgn}(D)$ is not a square in k. In the latter case, K is real quadratic over a quadratic extension of k, so we henceforth exclude this case. Elements in K have the form $\alpha = a + b\rho$ with $a, b \in k(t)$. The *conjugate* of α is $\overline{\alpha} = a - b\rho \in K$.

If g denotes the genus of K, then K is (an) *elliptic* (function field) if $g = 1$ and (a) *hyperelliptic* (function field) if $g > 1$. Then $\deg(D) = 2g+2$ if K is real and $\deg(D) = 2g+1$ if K is imaginary. If K is real, then we can "extract" a fixed square root $\sqrt{D}$ of D in the field $k\langle t^{-1}\rangle$ of Puiseux series over k, so $K \subseteq k\langle t^{-1}\rangle$. As a result, every nonzero element $\alpha \in K$ has a representation $\alpha = \sum_{i=-\infty}^{m} a_i t^i$ with $m \in \mathbb{Z}$, $a_i \in k$ for $i \leq m$, and $a_m \neq 0$. We set $\deg(\alpha) = m$ and $\lfloor \alpha \rfloor = \sum_{i=0}^{m} a_i t^i \in k[t]$. Also $\deg(0) = -\infty$ and $\lfloor 0 \rfloor = 0$.

Denote by $k[t]$ the ring of polynomials with coefficients in k in the variable t and let $\mathcal{O} = \overline{k[t]}$ be the integral closure of $k[t]$ in K. Then $\mathcal{O}$ is a $k[t]$-module of rank 2 with basis $\{1, \sqrt{D}\}$. An *(integral $\mathcal{O}$-)ideal* $\mathfrak{a}$ is a subset of $\mathcal{O}$ such that for any $\alpha, \beta \in \mathfrak{a}$ and $\theta \in \mathcal{O}$, $\alpha + \beta \in \mathfrak{a}$ and $\theta\alpha \in \mathfrak{a}$. A *fractional ($\mathcal{O}$-)ideal* $\mathfrak{a}$ is a subset of K such that $d\mathfrak{a}$ is an integral ideal for some nonzero $d \in k[t]$. Every fractional ideal $\mathfrak{a}$ is an $\mathcal{O}$-submodule of K. If the $\mathcal{O}$-rank of $\mathfrak{a}$ is 1, i.e. there exists $\alpha \in K$ such that $\mathfrak{a} = \{\theta\alpha \mid \theta \in \mathcal{O}\}$, then $\mathfrak{a}$ is *principal* and α is a *generator* of $\mathfrak{a}$; write $\mathfrak{a} = (\alpha)$.

Henceforth, all ideals (fractional and integral) are assumed to be nonzero, so the term "ideal" will always be synonymous with "nonzero ideal". Then every integral ideal $\mathfrak{a}$ is a $k[t]$-module of rank 2 with a $k[t]$-basis $\{SQ, SP + S\sqrt{D}\}$ where $S, Q, P \in k[t]$, $SQ \neq 0$, and Q divides $D^2 - P$. Write $\mathfrak{a} = (SQ, SP)$. We may assume that S and Q are monic and, after subtracting a suitable multiple of Q from P, that $\deg(P) < \deg(Q)$. Then S, Q, and P are unique. $\mathfrak{a}$ is *primitive* if $S = 1$. If $\mathfrak{a} = (Q, P)$ is primitive with Q monic and $\deg(P) < \deg(Q)$, then the pair (Q, P) is the *standard representation* of $\mathfrak{a}$ (see [32]; [38, 39] also use the term *adapted basis*), and $\mathfrak{a}$ is said to be in *standard form.* In practice, we require the degree inequality $\deg(P) < \deg(Q)$, but we do not insist on Q being monic. A primitive ideal $\mathfrak{a} = (Q, P)$ is *reduced* if $\deg(Q) \leq g$, the genus of K. Hence every reduced ideal can be uniquely represented by a pair of polynomials P, Q where Q is monic, Q divides $D - P^2$, and $\deg(P) < \deg(Q) \leq g$. This "small" representation makes reduced ideals very suitable for computation.

On the set $\mathcal{I}$ of (nonzero) fractional ideals of K, a multiplication is defined as follows. If $\mathfrak{a}$ and $\mathfrak{b}$ are fractional ideals, then the product $\mathfrak{a}\mathfrak{b}$ consists of all finite sums of products of the form $\alpha\beta$ with $\alpha \in \mathfrak{a}$ and $\beta \in \mathfrak{b}$. Under this multiplication, $\mathcal{I}$ is an infinite Abelian group with identity $\mathcal{O}$. The set $\mathcal{P}$ of (nonzero) fractional principal ideals is an infinite subgroup of $\mathcal{I}$ of finite index h', the *ideal class number* of K. The factor group $\mathcal{C} = \mathcal{I}/\mathcal{P}$ is the *ideal class group* of K. Note also that the set of integral ideals is a submonoid of $\mathcal{I}$. Two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ are *equivalent* if they lie in the same coset of $\mathcal{C}$, i.e. $\mathfrak{a} = (\theta)\mathfrak{b}$ for some $\theta \in K^*$. The element θ is a *relative generator* of $\mathfrak{a}$ with respect to $\mathfrak{b}$. Write $\mathfrak{a} \sim \mathfrak{b}$. Every equivalence class of ideals contains at least one and at most finitely many reduced ideals. If K is imaginary, then each class has a unique reduced representative [6];

however, if K is real, then there can be many reduced representatives in each ideal class, in fact, as many as $O(q^g)$ reduced ideals.

In the imaginary case, we base our cryptographic schemes on the arithmetic in the ideal class group $\mathcal{C}$ of K. Each ideal class is represented by its reduced representative. The product of two reduced ideals is generally not reduced; however, one can compute the reduced representative in the class of the product ideal quickly. Thus, the set of reduced ideals is a monoid under the following operation $*$: given two reduced ideals $\mathfrak{a}$ and $\mathfrak{b}$, let $\mathfrak{a} * \mathfrak{b}$ be the reduced ideal in the class of $\mathfrak{a}\mathfrak{b}$. The underlying discrete logarithm problem is the DLP in the class group $\mathcal{C}$: given reduced ideals $\mathfrak{d}$ and $\mathfrak{g}$ with $\mathfrak{d} \sim \mathfrak{g}^x$, find $x \pmod{h'}$.

Unfortunately, this approach fails in the real quadratic setting, due to the fact that there are many reduced representatives in each ideal class. Here, we restrict ourselves to the finite subset $\mathcal{R}$ of $\mathcal{P}$ of reduced principal ideals. More exactly, we define the *distance* $\delta(\mathfrak{a})$ of a reduced principal ideal $\mathfrak{a}$ to be the degree of a generator of minimal nonnegative degree. For $n \in \mathbb{N}_0$, we call $\mathfrak{a}$ the reduced principal ideal *below* n if $n - \delta(\mathfrak{a}) \geq 0$ and minimal. Then $\mathcal{R}$ is a monoid under the following operation: given two reduced principal ideals $\mathfrak{a}$ and $\mathfrak{b}$, let $\mathfrak{a} * \mathfrak{b}$ be the ideal below $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$. Here, the underlying DLP is the the following: given reduced principal ideals $\mathfrak{d}$ and $\mathfrak{g}$ so that $\mathfrak{d}$ is the reduced principal ideal below $x\delta(\mathfrak{g})$, find $x \pmod{R}$ where R is the maximal distance or the *regulator* of K. We will see that this problem is polynomially equivalent to the problem of finding the distance of a reduced principal ideal.

In both the imaginary and the real case, we require efficient algorithms for the following tasks:

- Given two reduced ideals in standard form, compute a standard representation of the product ideal.
- Given this product ideal, compute a reduced representative in its class.

In the real setting, we need to solve the additional problem:

- Given $n \in \mathbb{N}$, compute the reduced principal ideal below n.

3 Algorithms

In both the real and the imaginary models, the composition operation $*$, its implementation, and its complexity have previously been studied in considerable detail. For the imaginary setting, we refer the reader to [13], [31], and [32]. The real case is discussed in [38], [36], [34], and again [32]. A detailed complexity analysis with explicit O constants can be found in [32]. To make this paper somewhat self-contained, we restate the composition procedures here, but we only sketch proofs of correctness or performance.

Algorithms pertaining to just the imaginary setting have the prefix "I" in their name. Analogously, procedures that only apply to the real case begin with the letter "R". Algorithms with no prefix work in both types of representations. We use the complexity model of [32]; that is, all complexity estimates are stated in terms of elementary field operations of quadratic complexity, such as multiplication and inversion of field elements. Here, we assume standard complexity estimates for polynomial arithmetic as described for example on pp. 109f. of [14]; in particular, multiplication of two polynomials of respective degrees m and n $(m \geq n)$ requires $O(mn)$ field operations, division with remainder uses $O(n(m-n+1))$ field operations, and computation of extended gcd's takes $O(m^2)$ field operations (see also [32] for the last result).

Our first algorithm computes the standard representation of the product of two primitive ideals in standard form.

Algorithm MULT (ideal multiplication, real and imaginary case)

Input: (Q_a, P_a, Q_b, P_b) *where* $\mathfrak{a} = (Q_a, P_a)$ *and* $\mathfrak{b} = (Q_b, P_b)$ *are two primitive ideals.*

Output: (Q_c, P_c, S) *where* $\mathfrak{c} = (Q_c, P_c)$ *is a primitive ideal in standard form,* $S \in k[t]$*, and* $(S)\mathfrak{c} = \mathfrak{a}\mathfrak{b}$*. In the imaginary case,* S *need not be output.*

Algorithm:

1. $T \leftarrow \gcd(Q_a, Q_b) \leftarrow XQ_a \pmod{Q_b}$

   $(T, X \in k[x],\ \deg(X) < \deg(Q_b))$.

2. *If* $T = 1$*, then*

   $Y \leftarrow 1,\ Z \leftarrow 0,\ S \leftarrow 1$

   else

   $S \leftarrow \gcd(T, P_a + P_b) \leftarrow YT + Z(P_a + P_b) \quad (S, Y, Z \in k[x])$.

3. $Q_c \leftarrow \dfrac{Q_a Q_b}{S^2}$.

   $P_c \leftarrow P_a + \dfrac{Q_a XY(P_b - P_a) + Z(D - {P_a}^2)}{S} \pmod{Q_c}$

   $(\deg(P_c) < \deg(Q_c))$.

Proposition 3.1 *The parameters* $\mathfrak{c}$ *and* S *computed by Algorithm MULT satisfy* $(S)\mathfrak{c} = \mathfrak{a}\mathfrak{b}$*. Furthermore, if* $\mathfrak{a}$ *and* $\mathfrak{b}$ *are reduced, then* $\deg(S) < g$, $\deg(P_c) < \deg(Q_c) \leq 2g$*, and the algorithm performs* $O(g^2)$ *field operations.*

Proof: The correctness of the algorithm is proved in Section II.2 of [38] for the real case and follows from [13] for the imaginary case. The degree bounds follow from the fact that $\mathfrak{a}$ and $\mathfrak{b}$ are reduced and in standard form. For the complexity result, see Proposition 5 of [32]. □

Henceforth, we need to treat the imaginary and real settings separately. We begin with the imaginary situation and first describe the composition operation $*$ which computes the unique reduced representative in the class of the product ideal $\mathfrak{ab}$, where $\mathfrak{a}$ and $\mathfrak{b}$ are two reduced ideals. The procedure uses two different types of reduction steps. The first step is only used once at the beginning of the reduction process. If this does not produce a reduced ideal, the second step, which is computationally more efficient than the first step, is used subsequently, until a reduced ideal is obtained.

Algorithm I-RED-STEP1 (initial reduction step, imaginary case)

Input: (Q, P) *where* $\mathfrak{a} = (Q, P)$ *is a primitive ideal in standard form.*

Output: (Q_+, P_+, a, Q, P) *where* $\mathfrak{a}_+ = (Q_+, P_+)$ *is an ideal equivalent to* $\mathfrak{a}$ *in standard form and* $a = \lfloor -P/Q_+ \rfloor$.

Algorithm: $Q_+ \leftarrow \dfrac{D - P^2}{Q}, \quad a \leftarrow \left\lfloor \dfrac{-P}{Q_+} \right\rfloor, \quad P_+ \leftarrow -P - aQ_+$

(so $-P = aQ_+ + P_+$ *with* $\deg(P_+) < \deg(Q_+)$*.)*

Algorithm I-RED-STEP2 (subsequent reduction step, imaginary case)

Input: *The output* (Q, P, a_-, Q_-, P_-) *of I-RED-STEPi,* $i = 1$ *or 2.*

Output: (Q_+, P_+, a, Q, P) *where* $\mathfrak{a}_+ = (Q_+, P_+)$ *is an ideal equivalent to* $\mathfrak{a}$ *in standard form and* $a = \lfloor -P/Q_+ \rfloor$.

Algorithm: $Q_+ \leftarrow Q_- + a_-(P - P_-), \quad a \leftarrow \left\lfloor \dfrac{-P}{Q_+} \right\rfloor, \quad P_+ \leftarrow -P - aQ_+$

(so again $-P = aQ_+ + P_+$ *with* $\deg(P_+) < \deg(Q_+)$*.)*

We note that in both reduction steps, $\deg(Q_+) \leq \deg(Q) - 2$ (see [13, 32]).

Algorithm I-COMP (ideal composition, imaginary case)

Input: (Q_a, P_a, Q_b, P_b) *where* $\mathfrak{a} = (Q_a, P_a)$ *and* $\mathfrak{b} = (Q_b, P_b)$ *are two reduced ideals in standard form.*

Output: (Q_c, P_c) *where* $\mathfrak{c} = (Q_c, P_c) = \mathfrak{a}*\mathfrak{b}$ *is the reduced ideal equivalent to* $\mathfrak{ab}$ *in standard form.*

Algorithm:

1. $(Q_c, P_c) \leftarrow MULT(Q_a, P_a, Q_b, P_b)$.
2. *If* $\deg(Q_c) > g$ *then*
   - 2.1. $Q \leftarrow Q_c, \quad P \leftarrow P_c$.
   - 2.2. $(Q_c, P_c, a, Q, P) \leftarrow$ *I-RED-STEP1*(Q, P).
   - 2.3. *While* $\deg(Q_c) > g$

     $(Q_c, P_c, a, Q, P) \leftarrow$ *I-RED-STEP2*(Q_c, P_c, a, Q, P).

Proposition 3.2 *The ideal* $\mathfrak{c}$ *computed by Algorithm I-COMP is the reduced ideal equivalent to* $\mathfrak{ab}$ *in standard form. Furthermore, the algorithm performs* $O(g^2)$ *field operations.*

Proof: For the correctness of the algorithm, see [13]. By Theorem 1 of [32], the complexity of the procedure is $22g^2 + O(g)$. □

For our cryptographic schemes, we require an algorithm for "exponentiation" of reduced ideals. This method is based on the standard repeated squaring and multiplying technique used for ordinary exponentiation (see for example Algorithm 1.2.3, p. 9, of [14]).

Algorithm I-EXP (exponentiation, imaginary case)

Input: (Q_a, P_a, n) *where* $\mathfrak{a} = (Q_a, P_a)$ *is a reduced ideal in standard form and* $n \in \mathbb{N}_0$.

Output: (Q_b, P_b) *where* $\mathfrak{b} = (Q_b, P_b)$ *is the reduced ideal equivalent to* $\mathfrak{a}^n$ *in standard form.*

Algorithm:

1. *If* $n = 0$, *then*

   $Q_b \leftarrow 1, \quad P_b \leftarrow 0,$

   else
   - 1.1. *Compute the binary representation* $n = \sum_{i=0}^{l} b_i 2^{l-i}$ *of* n *where* $b_0 = 1$, $b_i \in \{0, 1\}$ *for* $1 \le i \le l$.

1.2. $Q_b \leftarrow Q_a$, $\quad P_b \leftarrow P_a$.

1.3. For $i \leftarrow 1$ *to* l *do*

1.3.1. $(Q_b, P_b) \leftarrow$ *I-COMP*(Q_b, P_b, Q_b, P_b).

1.3.2. If $b_i = 1$, *then* $(Q_b, P_b) \leftarrow$ *I-COMP*(Q_a, P_a, Q_b, P_b).

Proposition 3.3 *The ideal computed by Algorithm I-EXP is the reduced ideal equivalent to* $\mathfrak{a}^n$ *and the algorithm performs* $O(\max\{1,\ g^2 \log n\})$ *field operations.*

We now proceed with the real setting. For brevity, we set $d = \lfloor \sqrt{D} \rfloor$, i.e. d is the polynomial part of a (fixed) square root of D as defined in Section 2. As in the imaginary model, we have two reduction steps; the first one is to be applied immediately after ideal multiplication, while the second, more efficient one is for subsequent use.

If (Q, P) represents a basis of a reduced ideal $\mathfrak{a}$, then both reduction steps produce a *reduced representation* (Q_+, P_+) of an equivalent reduced ideal $\mathfrak{a}_+$; that is, $\deg(P_+ - d) < \deg(Q_+) < \deg(P_+ + d)$. Then $\deg(P_+) = \deg(d) = g + 1 > \deg(Q_+)$, and in fact d and P_+ agree in their two highest coefficients [38]. In the real setting, all the reduced ideals produced by our algorithms are in this *reduced form*, not in standard form. If a user prefers ideals in standard form, he or she can easily convert the reduced representation (Q, P) to a standard representation by replacing P by $P \pmod{Q}$, $\deg(P) < \deg(Q)$.

Algorithm R-RED-STEP1 (initial reduction step, real case)

Input: (Q, P) *where* $\mathfrak{a} = (Q, P)$ *is a primitive ideal.*

Output: (Q_+, P_+, a, r, Q, P) *where* $\mathfrak{a}_+ = (Q_+, P_+)$ *is a primitive ideal equivalent to* $\mathfrak{a}$ *and* $P + d = aQ + r$ *with* $\deg(r) < \deg(Q)$.

Algorithm:

1. $a \leftarrow \left\lfloor \dfrac{P+d}{Q} \right\rfloor$, $\quad r \leftarrow P + d \ (mod\ Q)$

(so $P + d = aQ + r$ *with* $\deg(r) < \deg(Q)$*).*

2. $P_+ \leftarrow d - r$, $\quad Q_+ \leftarrow \dfrac{D - P_+^2}{Q}$.

Algorithm R-RED-STEP2 (subsequent reduction step, real case)

Input: *The output* $(Q, P, a_-, r_-, Q_-, P_-)$ *of R-RED-STEPi,* $i = 1$ *or 2.*

Output: (Q_+, P_+, a, r, Q, P) *where* $\mathfrak{a}_+ = (Q_+, P_+)$ *is a primitive ideal equivalent to* $\mathfrak{a}$ *and* $P + d = aQ + r$ *with* $\deg(r) < \deg(Q)$.

Algorithm:

1. $a \leftarrow \left\lfloor \frac{P+d}{Q} \right\rfloor, \quad r \leftarrow P + d - aQ$
   (so again $P + d = aQ + r$ *with* $\deg(r) < \deg(Q)$*).*

2. $P_+ \leftarrow d - r, \quad Q_+ \leftarrow Q_- + a(r - r_-)$.

Since $(P_+ + \sqrt{D})\mathfrak{a} = (Q)\mathfrak{a}_+$, $\mathfrak{a}$ and $\mathfrak{a}_+$ are equivalent. For both reduction steps, we have $\deg(Q_+) < \deg(P_+) < \deg(Q)$, provided the input ideal is not reduced. So in this case, we again have $\deg(Q_+) \leq \deg(Q) - 2$. If the input ideal is reduced, then the output ideal is also reduced and in reduced representation; in particular $\deg(Q_+) < g + 1 = \deg(P_+)$.

Let $\mathfrak{a}_1, \mathfrak{a}_2, \ldots$ be a sequence of primitive ideals where $\mathfrak{a}_{i+1}$ is obtained by applying one of the reduction steps to $\mathfrak{a}_i$ ($i \in \mathbb{N}$). For $i \geq 0$, we associate with each ideal $\mathfrak{a}_{i+1} = (Q_i, P_i)$ the element $\alpha_i = (P_i + \sqrt{D})/Q_i \in K$. If we set

$$\theta_1 = 1, \qquad \theta_{i+1} = \prod_{j=1}^{i} \frac{1}{\alpha_j} \quad (i \in \mathbb{N}), \tag{3.1}$$

then $\mathfrak{a}_i = (\overline{\theta}_i)\mathfrak{a}_1$, where $\overline{\theta}_i$ is the conjugate of θ_i. Since $\alpha_i\overline{\alpha}_i = -Q_{i-1}/Q_i$, we have $\theta_i\overline{\theta}_i = (-1)^{i-1}Q_{i-1}/Q_0$ and hence for $i \in \mathbb{N}_0$:

$$\deg(\overline{\theta}_i) = \deg(Q_{i-1}) - \deg(Q_0) + \sum_{j=1}^{i-1} \deg(a_j) \tag{3.2}$$

where $a_j = \lfloor \alpha_j \rfloor$ for $j \in \mathbb{N}$.

The next algorithm shows how to obtain from any primitive ideal an equivalent reduced one. If the input ideal is not reduced, then we simply apply reduction steps until we obtain a reduced ideal (Q, P), i.e. $\deg(Q) \leq g$. Since each reduction step reduces the degree of Q by at least 2, this must eventually happen.

Algorithm R-REDUCE (ideal reduction, real case)

Input: (Q_a, P_a) *where* $\mathfrak{a} = (Q_a, P_a)$ *is a primitive ideal.*

Output: (Q_b, P_b, ϵ) *where* $\mathfrak{b} = (Q_b, P_b)$ *is a reduced ideal equivalent to* $\mathfrak{a}$ *in reduced form and* ϵ *is the degree of a relative generator of* $\mathfrak{b}$ *with respect to* $\mathfrak{a}$.

Algorithm:

1. $Q_b \leftarrow Q_a, \quad P_b \leftarrow P_a, \quad \epsilon \leftarrow 0.$

2. If $\deg(Q_b) > g$, *then*

2.1. $Q \leftarrow Q_a$.

2.2. $(Q_b, P_b, a, r, Q_a, P_a) \leftarrow$ *R-RED-STEP1*(Q_a, P_a).

2.3. While $\deg(Q_b) > g$ *do*

2.3.1. $(Q_b, P_b, a, r, Q_a, P_a) \leftarrow$ *R-RED-STEP2*$(Q_b, P_b, a, r, Q_a, P_a)$.

2.3.2. $\epsilon \leftarrow \epsilon + \deg(a)$,

2.4. $a \leftarrow \left\lfloor \dfrac{P_b + d}{Q_b} \right\rfloor, \quad \epsilon \leftarrow \epsilon + \deg(a) + \deg(Q_b) - \deg(Q)$.

Proposition 3.4 *The ideal* $\mathfrak{b}$ *computed by Algorithm R-REDUCE is a reduced ideal equivalent to* $\mathfrak{a}$ *in reduced form, and* ϵ *is the degree of a relative generator of* $\mathfrak{b}$ *with respect to* $\mathfrak{a}$.

Proof: For the proof that $\mathfrak{b}$ is a reduced ideal equivalent to $\mathfrak{a}$ in reduced form, see [38]. Suppose the reduction steps in the algorithm generate ideals $\mathfrak{a}_1 = \mathfrak{a}, \mathfrak{a}_2, \ldots, \mathfrak{a}_{i+1}$ where $\mathfrak{a}_j = (Q_{j-1}, P_{j-1})$ $(1 \le j \le i+1)$ and i is minimal with $\deg(Q_i) \le g$, so $\mathfrak{a}_{i+1}$ is the first reduced ideal in the sequence. Then we have $\epsilon = \sum_{j=1}^{i-1} \deg(a_j)$ at the end of the while loop, where $a_j = \lfloor (P_j + d)/Q_j \rfloor$. At the end of the algorithm, $\epsilon = \sum_{j=1}^{i} \deg(a_j) + \deg(Q_i) - \deg(Q_0)$ which is correct by (3.2). □

Let $\mathfrak{r}_1 = \mathcal{O}$ and recursively define $\mathfrak{r}_{i+1}$ to be the reduced principal ideal obtained by applying one of the two reduction steps to $\mathfrak{r}_i$. Since the set $\mathcal{R}$ of reduced principal ideals is finite and the reduced representation of a reduced ideal is unique, there exists $m \in \mathbb{N}$ such that applying a reduction step to $\mathfrak{r}_m$ yields $\mathfrak{r}_1$, so $m = |\mathcal{R}|$, the movement through $\mathcal{R}$ using reduction steps is periodic, and $m-1$ reduction steps applied to $\mathfrak{r}_1 = \mathcal{O}$ yields all of $\mathcal{R}$. This process also imposes a total order on $\mathcal{R} = \{\mathfrak{r}_1 = \mathcal{O}, \mathfrak{r}_2, \mathfrak{r}_3, \ldots, \mathfrak{r}_m\}$.

The *distance* of a reduced principal ideal $\mathfrak{r}_i$ $(1 \le i \le m)$ is $\delta(\mathfrak{r}_i) = \delta_i = \deg(\overline{\theta}_i)$ where θ_i is given by (3.1), so $\overline{\theta}_i$ is a generator of minimal nonnegative degree of $\mathfrak{r}_i$. Then the distance δ_i is a nonnegative function on $\mathcal{R}$ that strictly increases with i.

For two reduced principal ideals $\mathfrak{r}_i, \mathfrak{r}_j$ with $1 \le j \le i \le m$, the *(relative) distance* from $\mathfrak{r}_i$ to $\mathfrak{r}_j$ is $\delta(\mathfrak{r}_i, \mathfrak{r}_j) = \delta_i - \delta_j$. Let $\mathfrak{a}_1, \mathfrak{a}_i \in \mathcal{R}$, say $\mathfrak{a}_1 = \mathfrak{r}_j, \mathfrak{a}_i = \mathfrak{r}_{i-1+j}$ with $1 \le j < i \le m+1-j$. If $\mathfrak{a}_1 = (Q_0, P_0)$ and $\mathfrak{a}_i = (Q_{i-1}, P_{i-1})$, where these are reduced representations, then $\deg(a_{i-1}Q_{i-1}) = \deg(P_{i-1} + d) = g+1$ together with (3.2) implies

$$\delta(\mathfrak{a}_i, \mathfrak{a}_1) \;=\; g + 1 - \deg(Q_0) + \sum_{j=1}^{i-2} \deg(a_j). \tag{3.3}$$

In particular, for $j = 1$, we have $1 \le \deg(a_{i-1}) = g + 1 - \deg(Q_{i-1}) \le g$, so

$$\begin{array}{rcl rcl rcl} \delta_1 &=& 0, & \delta_2 &=& g+1, & \delta_{i+1} &=& \delta_i + \deg(a_{i-1}), \\ 1 &\le& \delta_{i+1} - \delta_i & \le & g, & & \delta_i &\ge& i \end{array} \tag{3.4}$$

for $i \ge 2$. For ease of notation, set $\delta_{m+1} = \deg(\overline{\theta}_{m+1})$ (strictly speaking, distances are only defined for indices up to m). $R = \delta_{m+1}$ is the *regulator* of K.

Let $k \in \mathbb{N}_0$, $k < R$. The reduced principal ideal *below* k is the unique reduced principal ideal $\mathfrak{r}_i$ with $\delta_i \le k < \delta_{i+1}$. If $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}$, then the ideal $\mathfrak{a} * \mathfrak{b}$ is the reduced principal ideal below $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$.

At times, we may need to advance from a reduced principal ideal a certain specified length in $\mathcal{R}$. For example, if we multiply two reduced principal ideals $\mathfrak{a}$ and $\mathfrak{b}$ using MULT and reduce the result using R-REDUCE, then we still need to perform reduction steps until we actually reach the ideal $\mathfrak{a} * \mathfrak{b}$ below $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$.

Our next four algorithms give as output a reduced principal ideal together with its 'error' ϵ in distance from the input. This error is always an integer between $-g$ and 0. Note, however, that the ideal distances themselves are never used.

Algorithm R-ADVANCE (advancement in $\mathcal{R}$, real case)

Input: (Q_a, P_a, k) *where* $\mathfrak{a} = (Q_a, P_a)$ *is a reduced principal ideal in reduced form and* $k \in \mathbb{N}_0$.

Output: (Q_b, P_b, ϵ) *where* $\mathfrak{b} = (Q_b, P_b)$ *is the reduced principal ideal below* $\delta(\mathfrak{a}) + k$ *in reduced form and* $\epsilon = \delta(\mathfrak{b}, \mathfrak{a}) - k$.

Algorithm:

1. $Q_b \leftarrow Q_a, \quad P_b \leftarrow P_a, \quad \epsilon = -k, \quad \delta \leftarrow g + 1 - \deg(Q_a) - k.$

2. If $\delta \le 0$, *then*

  2.1. $(Q_c, P_c, a, r, Q_b, P_b) \leftarrow$ *R-RED-STEP1*(Q_b, P_b).

  2.2. While $\delta \le 0$ *do*

    2.2.1. $\epsilon \leftarrow \delta$

    2.2.2. $(Q_c, P_c, a, r, Q_b, P_b) \leftarrow$ *R-RED-STEP2*$(Q_c, P_c, a, r, Q_b, P_b)$,

    2.2.3. $\delta \leftarrow \delta + \deg(a)$,

Proposition 3.5 *The quantities* $\mathfrak{b}$ *and* ϵ *computed by Algorithm R-ADVANCE are, respectively, the reduced principal ideal below* $\delta(\mathfrak{a}) + k$ *and the value of* $\delta(\mathfrak{b}, \mathfrak{a}) - k$. *Furthermore, the algorithm performs* $O(\max\{1, kg\})$ *field operations.*

Proof: In each step, the value of ϵ is $\delta(\mathfrak{b},\mathfrak{a}) - k$ by (3.3) where $\mathfrak{b} = (Q_b, P_b)$; also, $\delta = \delta(\mathfrak{c},\mathfrak{a}) - k$ where $\mathfrak{c}$ is one reduction step ahead of $\mathfrak{b}$. So the final ideal $\mathfrak{b}$ has maximal distance such that $\delta(\mathfrak{b},\mathfrak{a}) \leq k$ and is thus the reduced principal ideal below $\delta(\mathfrak{a}) + k$.

Now the degrees of P_a, Q_a, and d are all bounded by $g+1$, so step 2.1 requires $O(g^2)$ field operations. Since by (3.4), each reduction step advances the value of δ by at least 1, the loop in step 2.2 is executed at most k times (provided $k > 0$). Each reduction step in step 2.2.2 performs $O(g\deg(a))$ operations. By (3.4), the sum of the degrees of all the a values is $O(\delta(\mathfrak{b},\mathfrak{a})) = O(k)$. So the total number of operations performed in step 2 is $O(gk)$. □

We now have the tools to compute from any two reduced principal ideals $\mathfrak{a},\mathfrak{b}$ the reduced principal ideal $\mathfrak{a} * \mathfrak{b}$ below $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$. We again point out that the computation of $\mathfrak{a} * \mathfrak{b}$ does not require knowledge of $\delta(\mathfrak{a})$ or $\delta(\mathfrak{b})$.

Algorithm R-COMP (ideal composition, real case)

Input: (Q_a, P_a, Q_b, P_b) *where* $\mathfrak{a} = (Q_a, P_a)$ *and* $\mathfrak{b} = (Q_b, P_b)$ *are two reduced principal ideals.*

Output: (Q_c, P_c, ϵ) *where* $\mathfrak{c} = (Q_c, P_c) = \mathfrak{a} * \mathfrak{b}$ *is in reduced form and* $\epsilon = \delta(\mathfrak{a} * \mathfrak{b}) - \delta(\mathfrak{a}) - \delta(\mathfrak{b})$.

Algorithm:

1. $(Q_c, P_c, S) \leftarrow$ *MULT*(Q_a, P_a, Q_b, P_b).
2. $(Q_c, P_c, \delta) \leftarrow$ *R-REDUCE*(Q_c, P_c).
3. $(Q_c, P_c, \epsilon) \leftarrow$ *R-ADVANCE*$(Q_c, P_c, \deg(S) - \delta)$.

Proposition 3.6 *Algorithm R-COMP computes the reduced principal ideal* $\mathfrak{a} * \mathfrak{b}$ *below* $\delta(\mathfrak{a}) + \delta(\mathfrak{b})$ *and the quantity* $\epsilon = \delta(\mathfrak{a} * \mathfrak{b}) - \delta(\mathfrak{a}) - \delta(\mathfrak{b})$. *Furthermore, the algorithm performs* $O(g^2)$ *field operations.*

Proof: Let $\tilde{\mathfrak{c}}$ be the ideal generated by step 2. Then $\delta = \delta(\tilde{\mathfrak{c}}) - \delta(\mathfrak{a}) - \delta(\mathfrak{b}) + \deg(S)$. Step 3 computes the ideal $\mathfrak{c}$ below $\delta(\tilde{\mathfrak{c}}) + \deg(S) - \delta = \delta(\mathfrak{a}) + \delta(\mathfrak{b})$ which is $\mathfrak{a} * \mathfrak{b}$, and the correct value of ϵ. Since $0 \leq \deg(S) - \delta \leq 2g$ (see [38]), step 3 performs at most $2g$ reduction steps by (3.4). In fact, according to [32], the algorithm performs $21g^2 + O(g)$ field operations. □

The complexity results in the proofs of Propositions 3.2 and 3.6 show that composition in the real and the imaginary model perform at essentially the same speed, with the real setting being only slightly slower.

Using repeated applications of Algorithm R-COMP in combination with R-ADVANCE, we can adapt the binary exponentiation technique mentioned earlier to compute for a reduced principal ideal $\mathfrak{a}$ and an "exponent" $n \in \mathbb{N}$ the reduced ideal below $n\delta(\mathfrak{a})$.

Algorithm R-EXP (exponentiation, real case)

Input: (Q_a, P_a, n) *where* $\mathfrak{a} = (Q_a, P_a)$ *is a reduced principal ideal in reduced form and* $n \in \mathbb{N}_0$.

Output: (Q_b, P_b, ϵ) *where* $\mathfrak{b} = (Q_b, P_b)$ *is the reduced principal ideal below* $n\delta(\mathfrak{a})$ *in reduced form and* $\epsilon = \delta(\mathfrak{b}) - n\delta(\mathfrak{a})$.

Algorithm:

1. $\epsilon \leftarrow 0$.
2. *If* $n = 0$, *then*

   $Q_b \leftarrow 1, \quad P_b \leftarrow d,$

   else

   2.1. *Compute the binary representation* $n = \sum_{i=0}^{l} b_i 2^{l-i}$ *of* n *where* $b_0 = 1$, $b_i \in \{0, 1\}$ *for* $1 \leq i \leq l$.

   2.2. $Q_b \leftarrow Q_a, \quad P_b \leftarrow P_a$.

   2.3. *For* $i \leftarrow 1$ *to* l *do*

   2.3.1. $(Q_b, P_b, \delta) \leftarrow$ *R-COMP*(Q_b, P_b, Q_b, P_b).

   2.3.2. $(Q_b, P_b, \epsilon) \leftarrow$ *R-ADVANCE*$(Q_b, P_b, -(\delta + 2\epsilon))$.

   2.3.3. *If* $b_i = 1$, *then*

   2.3.3.1. $(Q_b, P_b, \delta) \leftarrow$ *R-COMP*(Q_a, P_a, Q_b, P_b).

   2.3.3.2. $(Q_b, P_b, \epsilon) \leftarrow$ *R-ADVANCE*$(Q_b, P_b, -(\delta + \epsilon))$.

Proposition 3.7 *The ideal computed by Algorithm R-EXP is the reduced principal ideal below* $n\delta(\mathfrak{a})$. *Furthermore, the algorithm performs* $O(\max\{1, g^2 \log n\})$ *field operations.*

Proof: We first observe that $(1, d)$ is the reduced representation of $\mathcal{O}$ which is the reduced principal ideal below $n\delta(\mathfrak{a})$ for $n = 0$. Assume now that $n > 0$ and set $s_0 = 1 = b_0$, $s_i = 2s_{i-1} + b_i$ for $1 \leq i \leq l$. Since $s_l = n$, it suffices to show that at the end of the

i-th iteration of the for loop, $\mathfrak{b}_i = (Q_b, P_b)$ is the reduced principal ideal below $s_i\delta(\mathfrak{a})$ and $\epsilon_i = \delta(\mathfrak{b}_i) - s_i\delta(\mathfrak{a})$ $(0 \le i \le l)$. For $i = 0$, we have $\mathfrak{b}_0 = \mathfrak{a}$ and $\epsilon_0 = 0$, which is correct. Now consider the $(i+1)$-st iteration of the for loop. Step 2.3.1 produces the reduced principal ideal $\mathfrak{b}$ below $2\delta(\mathfrak{b}_i)$ and $\delta = \delta(\mathfrak{b}) - 2\delta(\mathfrak{b}_i)$. Step 2.3.2 generates the reduced principal ideal $\tilde{\mathfrak{b}}$ below

$$\delta(\mathfrak{b}) - (\delta + 2\epsilon_i) = 2(\delta(\mathfrak{b}_i) - \epsilon_i) = 2s_i\delta(\mathfrak{a})$$

and

$$\tilde{\epsilon} = \delta(\tilde{\mathfrak{b}}) - \delta(\mathfrak{b}) + (\delta + 2\epsilon_i) = \delta(\tilde{\mathfrak{b}}) - 2s_i\delta(\mathfrak{a}).$$

If $b_{i+1} = 0$, then $s_{i+1} = 2s_i$ and $\mathfrak{b}_{i+1} = \tilde{\mathfrak{b}}$ which is correct. Suppose $b_{i+1} = 1$. Then step 2.3.3.1 computes the reduced principal ideal $\hat{\mathfrak{b}}$ below $\delta(\mathfrak{a}) + \delta(\tilde{\mathfrak{b}})$ and $\hat{\delta} = \delta(\hat{\mathfrak{b}}) - \delta(\mathfrak{a}) - \delta(\tilde{\mathfrak{b}})$. Finally, step 2.3.3.2 generates the reduced principal ideal $\mathfrak{b}_{i+1}$ below

$$\delta(\hat{\mathfrak{b}}) - (\hat{\delta} + \tilde{\epsilon}) = \delta(\mathfrak{a}) + \delta(\tilde{\mathfrak{b}}) - \tilde{\epsilon} = (1 + 2s_i)\delta(\mathfrak{a}) = s_{i+1}\delta(\mathfrak{a})$$

and

$$\epsilon_{i+1} = \delta(\mathfrak{b}_{i+1}) - \delta(\hat{\mathfrak{b}}) + (\hat{\delta} + \tilde{\epsilon}) = \delta(\mathfrak{b}_{i+1}) - s_{i+1}\delta(\mathfrak{a}).$$

Now $-g \le \delta, \epsilon \le 0$ for the values of δ and ϵ throughout the algorithm, so each R-ADVANCE call advances a distance of $O(g)$. Hence the complexity result follows from Propositions 3.6 and 3.5. □

Using the previous algorithm, we can now generate the reduced principal ideal below any nonnegative integer k. Here, we make use of the fact that if $\mathfrak{r}_1 = \mathcal{O} = (1, d)$ (in reduced form), then $\mathfrak{r}_2 = (D - d^2, d)$ with distance $\delta_2 = \deg(d) = g + 1$. If $k = n(g+1) + r$ with $r \le g$, then we first compute the reduced principal ideal $\mathfrak{a} = (Q_a, P_a)$ below $n(g+1)$ using R-EXP on the base ideal $\mathfrak{r}_2$ and the exponent n. Then we apply reduction steps until we reach the reduced principal ideal $\mathfrak{r} = (Q, P)$ below k.

Algorithm R-BELOW (generates an ideal of specific distance, real case)

Input: $k \in \mathbb{N}_0$.

Output: (Q, P, ϵ) *where* $\mathfrak{r} = (Q, P)$ *is the reduced principal ideal below* k *and* $\epsilon = \delta(\mathfrak{r}) - k$.

Algorithm:

1. $n \leftarrow \left\lfloor \dfrac{k}{g+1} \right\rfloor$, $r \leftarrow k - n(g+1)$ *(so* $k = n(g+1) + r$ *with* $r \le g$*).*
2. $(Q, P, \delta) \leftarrow$ *R-EXP*$(D - d^2, d, n)$.

3. $(Q, P, \epsilon) \leftarrow$ *R-ADVANCE*$(Q, P, r - \delta)$.

Proposition 3.8 *Algorithm R-BELOW computes the reduced principal ideal* $\mathfrak{r}$ *below* k *and the quantity* $\epsilon = \delta(\mathfrak{r}) - k$. *Furthermore, the algorithm performs* $O(g^2 \log k)$ *field operations.*

Proof: Let $\mathfrak{a} = (Q_a, P_a)$ be the output ideal of step 2. Then

$$\delta = \delta(\mathfrak{a}) - n(g+1) = \delta(\mathfrak{a}) - k + r$$

and $\epsilon = \delta(\mathfrak{r}) - \delta(\mathfrak{a}) - (r - \delta) = \delta(\mathfrak{r}) - k$. Step 2 requires $O(g^2 \log n) = O(g^2 \log k)$ field operations. Since $0 \le r - \delta \le 2g$, step 3 requires $O((r-\delta)g) = O(g^2)$ field operations. $\square$

4 Cryptographic Schemes

We now present three cryptographic schemes for both real and imaginary of quadratic function fields, namely a key exchange protocol, a public-key system, and signature scheme. Each system uses the "exponentiation" method corresponding to the composition operation. The field K should be chosen so that q^g is large. More details on the choice of the field are given in the next section. For simplicity, we assume $D(t)$ to be monic.

We begin with the imaginary case. All our schemes require the following precomputation.

I-PRECOMP (precomputation, imaginary case)

1. *Generate an odd prime power* q,
2. *generate a random squarefree monic polynomial* $D \in \mathbb{F}_q[t]$ *of odd degree,*
3. *generate a random ideal* $\mathfrak{i} = (Q_i, P_i)$ *and compute* $\mathfrak{r} = (Q, P) \leftarrow$ *I-REDUCE*(Q_i, P_i),
4. *publicize* (q, D, Q, P).

KEY EXCHANGE PROTOCOL (imaginary case)

Precomputation:

Alice and Bob jointly obtain (q, D, Q, P) *by performing I-PRECOMP.*

Protocol:

1. *Alice*

1.1. secretly generates an integer a, $0 < a < q^{\lfloor g/2 \rfloor}$,

1.2. computes $(Q_a, P_a) \leftarrow$ *I-EXP*(Q, P, a),

1.3. transmits (Q_a, P_a) *to Bob.*

2. Bob

2.1. secretly generates an integer b, $0 < b < q^{\lfloor g/2 \rfloor}$,

2.2. computes $(Q_b, P_b) \leftarrow$ *I-EXP*(Q, P, b),

2.3. transmits (Q_b, P_b) *to Alice.*

3. Alice computes $(Q_k, P_k) \leftarrow$ *I-EXP*(Q_b, P_b, a).

4. Bob computes $(Q_k, P_k) \leftarrow$ *I-EXP*(Q_a, P_a, b).

Since $(Q_b, P_b)^a \sim ((Q, P)^b)^a \sim (Q_k, P_k) \sim ((Q, P)^a)^b \sim (Q_a, P_a)$, both parties have now computed the unique reduced ideal equivalent to $\mathfrak{r}^{ab}$ in standard form. They can use the polynomials (Q_k, P_k) (or any previously agreed upon portion thereof) as their key. Since $\mathfrak{a}$ and $\mathfrak{b}$ are reduced and in standard form, both parties transmit approximately $2g \log q$ bits of information.

Denote by $x \oplus y$ the bitwise "exclusive or" of two bit strings x and y.

PUBLIC-KEY CRYPTOSYSTEM (imaginary case)

Key Generation:

All participants jointly obtain (q, D, Q, P) *by performing I-PRECOMP.*

Each participant

1. secretly generates an integer a, $0 < a < q^{\lfloor g/2 \rfloor}$,

2. computes $(Q_a, P_a) \leftarrow$ *I-EXP*(Q, P, a),

3. makes (Q_a, P_a) *the public key and* a *the secret key.*

Encryption: *To encrypt a message* M, *the sender (with secret key* s*)*

1. looks up the recipient's public key (Q_r, P_r),

2. computes $(Q_k, P_k) \leftarrow$ *I-EXP*(Q_r, P_r, s),

3. repeat

3.1. generates the bit string $x \in \mathbb{N}$ *obtained by concatenating the coefficients in* $\mathbb{F}_q$ *of the polynomial* Q_k *written in binary,* *3.2. removes the first block m with $m < x$ from M,*

3.3. sends the ciphertext $m \oplus x$,

until all of M encrypted.

Decryption: *To decrypt a ciphertext C, the recipient (with secret key r)*

1. *looks up the sender's public key (Q_s, P_s),*
2. *computes $(Q_k, P_k) \leftarrow$ I-EXP(Q_s, P_s, r),*
3. *repeat*

   3.1. generates the bit string $x \in \mathbb{N}$ obtained by concatenating the coefficients in $\mathbb{F}_q$ of the polynomial Q_k written in binary,

   3.2. removes the first block c with $c < x$ from C,

   3.3. computes $m \leftarrow c \oplus x$,

   until all of C is decrypted,
4. *concatenates all the blocks m to obtain the plaintext M.*

Once again $(Q_s, P_s)^r \sim (Q, P)^{sr} \sim (Q_r, P_r)^s$, so both parties compute the same reduced ideal (Q_k, P_k) and thus the same polynomial Q_k and bit string x. Since $\deg(Q_k) \leq g$, each block c of ciphertext is approximately $g \log q$ bits long.

Our next scheme requires a collision-resistant one-way hash function *hash* that takes as input a message M and the polynomials Q and P of a reduced ideal (in standard form) and produces positive integer values not exceeding $q^{\lfloor g/2 \rfloor}$. The inputs of this hash function can also be thought of as bit strings, if we concatenate M, Q, and P. The idea of using a hash function as described above was first presented in [29], where it was used for a signature scheme based on a real quadratic function field of characteristic 2.

For $n \in \mathbb{N}_0$, we let $\mathbb{F}_q[t]^{\leq n}$ denote the set of polynomials in $\mathbb{F}_q[t]$ of degree $\leq n$.

SIGNATURE SCHEME (imaginary case)

Precomputation: *All participants jointly*

1. *obtain (q, D, Q, P) by performing I-PRECOMP.*
2. *agree on a cryptographically secure hash function*

   $hash : \mathbb{N} \times \mathbb{F}_q[t]^{\leq g} \times \mathbb{F}_q[t]^{\leq g-1} \to \{1, 2, \ldots, q^{\lfloor g/2 \rfloor}\}$

Each participant

1. *secretly generates an integer* a*,* $0 < a < q^{\lfloor g/2 \rfloor}$*,*

2. *computes* $(Q_a, P_a) \leftarrow$ *I-EXP*(Q, P, a)*,*

3. *makes* (Q_a, P_a) *the public key and* a *the secret key.*

Signature Generation: *To sign a message* M*, the signer (with secret key* a*)*

1. *secretly generates an integer* r*,* $0 < r < q^{\lfloor g/2 \rfloor}$*,*

2. *computes* $(Q_r, P_r) \leftarrow$ *I-EXP*(Q, P, r)*,*

3. *computes* $m \leftarrow hash(M, Q_r, P_r)$*,*

4. *computes* $s \leftarrow r - ma$*,*

5. *sends the signature* (Q_r, P_r, s) *along with the message* M*.*

Signature Verification: *To verify the signature* (Q_r, P_r, s) *to the message* M*, the verifier*

1. *computes* $m \leftarrow hash(M, Q_r, P_r)$*,*

2. *computes* $(Q_s, P_s) \leftarrow$ *I-EXP*$(Q, P, |s|)$*,*

3. *looks up the senders public key* (Q_a, P_a)*,*

4. *computes* $(Q_b, P_b) \leftarrow$ *I-EXP*(Q_a, P_a, m)*,*

5. *computes* $(Q_{r'}, P_{r'}) \leftarrow$ *I-COMP*$(Q_b, P_b, Q_s, sgn(s)P_s)$*,*

6. *accepts the signature if and only if* $Q_{r'} = Q_r$ *and* $P_{r'} = P_r$*.*

To show that the verification procedure is correct, we first observe that if $(\tilde{Q}, \tilde{P})$ is any reduced ideal, then $(\tilde{Q}, -\tilde{P})$ is the reduced representative of the ideal class that is the inverse in the class group $\mathcal{C}$ of the ideal class of $(\tilde{Q}, \tilde{P})$: it is clear that $(\tilde{Q}, -\tilde{P})$ is reduced and in standard form, and multiplying $(\tilde{Q}, \tilde{P})$ and $(\tilde{Q}, -\tilde{P})$ using algorithm MULT gives as result the ideal $\mathcal{O} = (1, 0)$.

Now we have $(Q_s, \mathrm{sgn}(P_s)P_s) \sim (Q, P)^s$ and $(Q_b, P_b) \sim (Q_a, P_a)^m \sim (Q, P)^{am}$. If $s \geq 0$, then $(Q_{r'}, P_{r'}) = (Q_b, P_b) * (Q_s, P_s) \sim (Q, P)^{am+s} \sim (Q, P)^r \sim (Q_r, P_r)$. If $s < 0$, then $(Q_s, P_s) \sim (Q, P)^{-s}$ and $(Q_{r'}, P_{r'}) = (Q_b, P_b) * (Q_s, -P_s) \sim (Q, P)^{am+s} \sim (Q_r, P_r)$.

To forge a signature, an adversary needs to generate a reduced ideal $\mathfrak{r} = (Q_r, P_r)$ and an integer s such that $(Q, P)^s \sim (Q_r, P_r) * (Q_b, -P_b)$. It is necessary to use the hash function to prevent the following attack. Suppose we replaced $m = \text{hash}(M, Q_r, P_r)$ by a message block m where $0 < m < q^{\lfloor g/2 \rfloor}$. Then an adversary can simply pick a random positive integer s and compute the reduced ideal (Q_r, P_r) equivalent to $(Q, P)^s * (Q_b, P_b)$. Such a forged signature (Q_r, P_r, s) would always be accepted by the verifier. The hash function forces the signer to generate the reduced ideal (Q_r, P_r) *before* computing s, rather than making it possible to choose s first and then generate a "fitting" reduced ideal (Q_r, P_r).

Since $0 < a, m, r < g^{g/2}$, we have $-q^{g/2} < s < q^g$, so the transmission of a signature requires approximately $3g \log q$ bits of information. Signatures can be somewhat shortened by imposing a smaller upper bound on a and m.

We now continue with cryptographic schemes in the real setting. Some of the schemes require a participant to generate a reduced principal ideal together with its distance. The easiest way to achieve this is to generate a random nonnegative integer a of desired size and compute $(Q_a, P_a, \epsilon_a) \leftarrow$ R-BELOW(a). Then $\delta(\mathfrak{a}) = a + \epsilon_a$. The schemes also require the precomputation of $d = \lfloor \sqrt{D} \rfloor$. This can be done using a Puiseux diagram (see [25]). Once again, there is a common precomputation to all schemes.

R-PRECOMP (precomputation, real case)

1. *Generate an odd prime power q,*
2. *generate a random squarefree monic polynomial $D \in \mathbb{F}_q[t]$ of even degree,*
3. *compute $d \leftarrow \lfloor \sqrt{D} \rfloor$,*
4. *publicize (q, D, d).*

The protocol given below is a slight improvement over the versions given in [36] and [29] in that it eliminates the need for including a reduced ideal in the set of public parameters.

KEY EXCHANGE PROTOCOL (real case)

Precomputation:

Alice and Bob jointly obtain (q, D, d) by performing R-PRECOMP.

Protocol:

1. *Alice*
   1.1. *secretly generates a reduced principal ideal $\mathfrak{a} = (Q_a, P_a)$ with distance $\delta(\mathfrak{a}) < q^{\lfloor g/2 \rfloor}$,*
   1.2. *transmits (Q_a, P_a) to Bob.*

2. *Bob*
   2.1. *secretly generates a reduced principal ideal* $\mathfrak{b} = (Q_b, P_b)$ *with distance* $\delta(\mathfrak{b}) < q^{\lfloor g/2 \rfloor}$,
   2.2. *transmits* (Q_b, P_b) *to Alice.*
3. *Alice computes* $(Q_k, P_k) \leftarrow$ *R-EXP*$(Q_b, P_b, \delta(\mathfrak{a}))$.
4. *Bob computes* $(Q_k, P_k) \leftarrow$ *R-EXP*$(Q_a, P_a, \delta(\mathfrak{b}))$.

Both parties compute the reduced principal ideal (Q_k, P_k) below $\delta(\mathfrak{a})\delta(\mathfrak{b})$. They can use Q_k and P_k (or any previously agreed upon portion thereof) as their key. As in the imaginary case, both parties transfer approximately $2g \log q$ bits of information.

PUBLIC KEY CRYPTOSYSTEM (real case)

Key Generation:

All participants jointly obtain (q, D, d) *by performing R-PRECOMP.*

Each participant

1. *secretly generates a reduced principal ideal* $\mathfrak{a} = (Q_a, P_a)$ *with distance* $\delta(\mathfrak{a}) < q^{\lfloor g/2 \rfloor}$,
2. *makes* (Q_a, P_a) *the public key and* $\delta(\mathfrak{a})$ *the secret key.*

Encryption: *To encrypt a message* M*, the sender (with secret key* $\delta(\mathfrak{s})$*)*

1. *looks up the recipient's public key* (Q_r, P_r),
2. *computes* $(Q_k, P_k, \epsilon) \leftarrow$ *R-EXP*$(Q_r, P_r, \delta(\mathfrak{s}))$,
3. *repeat*
   3.1. *generates the bit string* $x \in \mathbb{N}$ *obtained by concatenating the coefficients in* $\mathbb{F}_q$ *of the polynomial* Q_k *written in binary,*
   3.2. *removes the first block* m *with* $m < x$ *from* M,
   3.3. *sends the ciphertext* $m \oplus x$,

   until all of M *is encrypted.*

Decryption: *To decrypt a ciphertext* C*, the recipient (with secret key* $\delta(\mathfrak{r})$*)*

1. *looks up the sender's public key* (Q_s, P_s),
2. *computes* $(Q_k, P_k, \epsilon) \leftarrow R - EXP(Q_s, P_s, \delta(\mathfrak{r}))$,

3. *repeat*
   3.1. *generates the bit string $x \in \mathbb{N}$ obtained by concatenating the coefficients in $\mathbb{F}_q$ of the polynomial Q_k written in binary,*
   3.2. *removes the first block c with $c < x$ from C,*
   3.3. *computes $m \leftarrow c \oplus x$,*

   until all of C is decrypted,
4. *concatenates all the blocks m to obtain the plaintext M.*

Both parties compute the reduced principal ideal (Q_k, P_k) below $\delta(\mathfrak{r})\delta(\mathfrak{s})$ and thus the same polynomial Q_k and bit string x. Again each block c of ciphertext is approximately $g \log q$ bits long.

Our next scheme is an improvement over the signature scheme of [29] in that it generates shorter signatures. Once again, we use a cryptographically secure hash function similar to the one used in the corresponding scheme in imaginary fields (the signature scheme of [34] failed to take this into consideration).

SIGNATURE SCHEME (real case)

Precomputation:

All participants jointly

1. *obtain (q, D, d) by performing R-PRECOMP.*
2. *agree on a cryptographically secure hash function*
   $hash : \mathbb{N} \times \mathbb{F}_q[t]^{\leq g} \times \mathbb{F}_q[t]^{\leq g+1} \rightarrow \{1, 2, \ldots, q^{\lfloor g/2 \rfloor}\}$.

Each participant

1. *generates a reduced principal ideal $\mathfrak{a} = (Q_a, P_a)$ with distance $\delta(\mathfrak{a}) < q^{\lfloor g/2 \rfloor}$,*
2. *makes (Q_a, P_a) the public key and $\delta(\mathfrak{a})$ the secret key.*

Signature Generation: *To sign a message M, the signer (with secret key $\delta(\mathfrak{a})$)*

1. *secretly generates a reduced principal ideal $\mathfrak{r}$ with distance $\delta(\mathfrak{r}) < q^{\lfloor g/2 \rfloor}$.*
2. *computes $m \leftarrow \text{hash}(M, Q_r, P_r)$,*
3. *computes $s \leftarrow \delta(\mathfrak{r}) - m\delta(\mathfrak{a})$,*

4. sends the signature (Q_r, P_r, s) along with the message M.

Signature Verification: *To verify the signature (Q_r, P_r, s) to the message M, the verifier*

1. *computes $m \leftarrow$ $\text{hash}(M, Q_r, P_r)$,*
2. *computes $(Q_s, P_s, \delta) \leftarrow BELOW(|s|)$,*
3. *looks up the sender's public key (Q_a, P_a),*
4. *computes $(Q_b, P_b, e) \leftarrow$ R-EXP(Q_a, P_a, m),*
5. *if $s \geq 0$ then*
   5.1. *computes $(Q_c, P_c, f) \leftarrow$ R-COMP(Q_s, P_s, Q_b, P_b),*
   5.2. *computes $(Q_{r'}, P_{r'}, \epsilon) \leftarrow$ R-ADVANCE$(Q_c, P_c, -(\delta + e + f))$,*
   5.3. *accepts the signature if and only if $Q_{r'} = Q_r$ and $P_{r'} = P_r$.*
   else /$s < 0$*/*
   5.4. *computes $(Q_c, P_c, f) \leftarrow$ R-COMP(Q_r, P_r, Q_s, P_s),*
   5.5. *computes $(Q_{b'}, P_{b'}, e') \leftarrow$ R-ADVANCE$(Q_c, P_c, -(\delta + f))$,*
   5.6. *accepts the signature if and only if $Q_{b'} = Q_b$ and $P_{b'} = P_b$.*

Once again, we check the verification procedure. Write $\mathfrak{s} = (Q_s, P_s)$, $\mathfrak{b} = (Q_b, P_b)$, $\mathfrak{c} = (Q_c, P_c)$, $\mathfrak{r}' = (Q_{r'}, P_{r'})$, and $\mathfrak{b}' = (Q_{b'}, P_{b'})$. Suppose first that $s \geq 0$. Then we have

$$\delta + e + f = (\delta(\mathfrak{s}) - s) + (\delta(\mathfrak{b}) - m\delta(\mathfrak{a})) + (\delta(\mathfrak{c}) - \delta(\mathfrak{s}) - \delta(\mathfrak{b})) = \delta(\mathfrak{c}) - \delta(\mathfrak{r}).$$

$\mathfrak{r}'$ is the reduced principal ideal below $\delta(\mathfrak{c}) - (\delta + e + f) = \delta(\mathfrak{r})$, so $\mathfrak{r}' = \mathfrak{r}$. Now assume that $s < 0$. Then

$$\delta + f = (\delta(\mathfrak{s}) + s) + (\delta(\mathfrak{c}) - \delta(\mathfrak{r}) - \delta(\mathfrak{s})) = \delta(\mathfrak{c}) - m\delta(\mathfrak{a}).$$

$\mathfrak{b}'$ is the reduced principal ideal below $\delta(\mathfrak{c}) - (\delta + f) = m\delta(\mathfrak{a})$, so $\mathfrak{b}' = \mathfrak{b}$.

To forge a signature (Q_r, P_r, s), an opponent must generate a reduced principal ideal $\mathfrak{r} = (Q_r, P_r)$ and an integer s with the following properties. If $s \geq 0$, then $\mathfrak{r}$ is the reduced principal ideal below $s + m\delta(\mathfrak{a})$. If $s < 0$, then $\mathfrak{b}$ is the reduced principal ideal below $\delta(\mathfrak{r}) - s$. The similarities in these two cases are best seen as follows. Let $(\mathfrak{s}, \delta)$ be as in step 2 of the verification procedure. If $s \geq 0$, write $\delta(\mathfrak{r}) = s + m\delta(\mathfrak{a}) = \delta(\mathfrak{s}) + \delta(\mathfrak{b}) - (\delta + e)$, so $\delta(\mathfrak{s}) + \delta(\mathfrak{b}) = \delta(\mathfrak{r}) + (\delta + e)$. Since $-2g \leq \delta + e \leq 0$, s must be such that $\mathfrak{r}$ has distance at most $2g$ below $\mathfrak{s} * \mathfrak{b}$. If $s < 0$, write $\delta(\mathfrak{b}) - e = m\delta(\mathfrak{a}) = \delta(\mathfrak{r}) + |s| = \delta(\mathfrak{r}) + \delta(\mathfrak{s}) - \delta$, so

$\delta(\mathfrak{r}) + \delta(\mathfrak{s}) = \delta(\mathfrak{b}) + (\delta - e)$. Since $-g \leq \delta - e \leq g$, s must be such that $\mathfrak{b}$ has distance within g of $\mathfrak{r} * \mathfrak{s}$.

Since $m, \delta(\mathfrak{a}), \delta(\mathfrak{r}) < q^{g/2}$, we have $-q^g < s < q^{g/2}$, so the transmission of a signature again requires at most $3g \log q$ bits of information. The signatures in [29] are between $3g \log q$ and $4g \log q$ bits long. Once again, signatures could be shortened by imposing tighter bounds on $\delta(\mathfrak{a})$ and m.

5 Security

Before we discuss possible attacks on our cryptographic schemes, we explore the size of our underlying sets. We need to ensure that the class group $\mathcal{C}$ (in the imaginary setting) and the set $\mathcal{R}$ of reduced principal ideals (in the real setting) are sufficiently large.

The order of $\mathcal{C}$ is simply the ideal class number h'. Consider now the real case and let $m = |\mathcal{R}|$. From (3.3) and (3.4)

$$m + 1 \leq R = g + 1 + \sum_{j=1}^{m-1} \deg(a_j) \leq g + 1 + g(m-1) = gm + 1.$$

where $R = \delta_{m+1}$ is the regulator of K. So the size of $\mathcal{R}$ is determined by R (in the elliptic case, we even have $R = m + 1$). Let h be the order of the *Jacobian* $\mathcal{J}$ of the curve C defining the function field K. Then

$$h = \begin{cases} h' & \text{in the imaginary case} \\ Rh' & \text{in the real case} \end{cases}$$

(see [42]). It is well-known (see for example Theorem V.1.15, p. 166, of [41]) that $h = L(1)$ where $L(x) \in \mathbb{Z}[x]$ is the *L polynomial* of $K|k$. Here, $L(x) = \prod_{i=1}^{2g}(1 - \alpha_i x)$ where each α_i $(1 \leq i \leq 2g)$ is an algebraic integer of absolute value $\sqrt{q}$ by the Hasse-Weil Theorem (see Theorem V.2.1, p. 169, of [41]). It follows that $(\sqrt{q} - 1)^{2g} \leq h \leq (\sqrt{q} + 1)^{2g}$ and hence

$$h = q^g + \Theta(q^{g-0.5}). \tag{5.1}$$

This implies that in an imaginary quadratic field, there are approximately q^g ideal classes in $\mathcal{C}$. Analyzing the size of $\mathcal{R}$ in the real case is slightly more complicated. To ensure a large regulator, we need to make h' as small as possible. A strong heuristic argument ([36], also Section 3.4.1, pp. 107-111, of [39]), analogous to the Cohen-Lenstra heuristic in real quadratic number fields [15, 16]), shows that the probability that the order of the odd part of the ideal class group exceeds x is $1/2x + O((\log x)/x^2)$. In fact, in the elliptic case, there is very strong numerical evidence that ideal class groups behave according to this heuristic [21, 22], and it can be proved that for sufficiently large q, the probability that $h' = 1$ is high [23]. At the same time, it is easy to find real quadratic fields whose ideal class number is odd; for example, by a result of Zhang [43], it suffices to choose D to be irreducible over k or the product of two odd-degree irreducibles in $k[t]$. Hence under these choices, h' is small with high probability, and there are close to q^g reduced principal ideals in K.

Hence to foil an exhaustive search attack, we should ensure that q^g is sufficiently large. Considerations for good choices of q and g are discussed below, but we point out that within these considerations, users can take advantage of the following trade-off. For small g and large q, our complexity analysis in section 3 results in very good performance in terms of field operations, but field arithmetic dominates our computation times. If q is small and g is large, then field arithmetic is very fast, but the number of field operations performed by our algorithms is high. Thus, one could select q and g in such a way as to optimize performance, while ensuring a sufficient level of security in our systems.

We now explore the possibility of breaking our schemes. We begin with the imaginary model. Here, the relevant problem underlying all three schemes is the DLP in the class group $\mathcal{C}$ of K: given two ideals $\mathfrak{g}$ and $\mathfrak{d}$ with $\mathfrak{d} \sim \mathfrak{g}^x$ for some $x \in \{0, 1, \ldots, h'-1\}$, find the discrete logarithm x. It is obvious that for any of the schemes, there is a polynomial-time reduction from an algorithm for solving the DLP to an algorithm for breaking the system. Since no other way of compromising any of the schemes is known, we focus our attention on the difficulty of the DLP in $\mathcal{C}$.

The ideal class group $\mathcal{C}$ is isomorphic to the Jacobian $\mathcal{J}$, and the DLPs in $\mathcal{C}$ and $\mathcal{J}$ are polynomially equivalent. We first observe that in certain cases, the DLP in the Jacobian of an elliptic or hyperelliptic curve is reducible to the DLP in a finite field, in which case using (hyper)elliptic function fields represents no advantage over using finite fields for the implementation of discrete logarithm based cryptosystems. More exactly, the curve C should not be supersingular [27], and the largest prime divisor of h' should not divide $q^k - 1$ for those small k for which the DLP in $\mathbb{F}_{q^k}$ is feasible [20]. It is currently unknown whether such reductions are possible in situations other than those cited above.

In [4], a probabilistic algorithm for computing discrete logarithms in $\mathcal{J}$ in the case where q is a prime is given. This technique is subexponential of complexity $\exp(c\sqrt{\log q^g \log\log q^g})$ where $c > 0$ is a constant, provided $\log q \leq (2g+1)^{.98}$, i.e. q is small compared to g. This algorithm was generalized to odd prime powers q in [7, 19], but seems infeasible in practice for sufficiently large parameters, and one could foil an attack based on this method by choosing q to be large and the genus g to be small.

In general, a technique analogous to that of Pohlig-Hellman [33] can be used to compute discrete logarithms in $\mathcal{C}$. The complexity of this method is essentially of order $\sqrt{p}$

where p is the largest prime factor of h'. This attack requires that h' be known. A technique described in [26] can be used to compute h' by generating the coefficients of the L polynomial $L(x)$; this method works particularly well for small g. Another algorithm for computing (among other quantities) h' given in [3] is polynomial in the size of q and exponential in g. Hence, while it might be feasible to determine h', particularly for small g, the Pohlig-Hellman attack is infeasible unless h' is smooth, i.e. has only small prime factors.

In the real setting, there are two problems that are relevant to possible attacks on our cryptoschemes. The *distance problem* (DP) in $\mathcal{R}$ requires the computation of the distance of a reduced principal ideal. The DLP in $\mathcal{R}$ is the problem of finding $x \pmod R$, given reduced principal ideals $\mathfrak{g}$ and $\mathfrak{d}$ where $\mathfrak{d}$ is the reduced principal ideal below $x\delta(\mathfrak{g})$. Both problems are equally difficult; hence, the problem of breaking any of our schemes is polynomial-time reducible to either problem.

Proposition 5.1 *There is a polynomial-time reduction from the DLP to the DP and vice versa.*

Proof: Suppose first that we can solve any instance of the DLP. Let $\mathfrak{r}$ be a reduced principal ideal. We wish to find $\delta(\mathfrak{r})$. Let $\delta(\mathfrak{r}) = y(g+1) + r$ with $0 \le r \le g$. Let $\mathfrak{r}'$ be the ideal below $(y+2)\delta_2$ where as before, $\delta_2 = \delta(\mathfrak{r}_2) = g+1$ and $\mathfrak{r}_2 = (D - d^2, d)$. Then

$$\begin{array}{llllll} \delta(\mathfrak{r}') & \le & (y+2)(g+1) & = \delta(\mathfrak{r}) - r + 2(g+1) & \le & \delta(\mathfrak{r}) + 2(g+1), \\ \delta(\mathfrak{r}') & > & (y+2)(g+1) - g & = \delta(\mathfrak{r}) - r + g + 2 & \ge & \delta(\mathfrak{r}) + 2. \end{array}$$

Suppose the call R-ADVANCE$(Q_r, P_r, 3g+1)$ with $\mathfrak{r} = (Q_r, P_r)$ generates a sequence $\mathfrak{a}_1 = \mathfrak{r}, \mathfrak{a}_2, \ldots, \mathfrak{a}_m$ of reduced principal ideals which we store in memory. Then $3g+1 \ge \delta(\mathfrak{a}_m, \mathfrak{a}_1) > (3g+1) - g = 2g+1$, so

$$\begin{array}{llllllll} \delta(\mathfrak{a}_1) & = & \delta(\mathfrak{r}) & < & \delta(\mathfrak{r}') - 2 & & < & \delta(\mathfrak{r}'), \\ \delta(\mathfrak{a}_m) & \ge & \delta(\mathfrak{r}) + 2g + 2 & \ge & \delta(\mathfrak{r}') - 2(g+1) + (2g+2) & & = & \delta(\mathfrak{r}'), \end{array}$$

hence $\delta(\mathfrak{a}_1) < \delta(\mathfrak{r}') \le \delta(\mathfrak{a}_m)$ and $\mathfrak{r}' \in \{\mathfrak{a}_2, \mathfrak{a}_3, \ldots, \mathfrak{a}_m\}$. Using the $m-1$ candidates $\mathfrak{a}_i$ $(2 \le i \le m)$ for $\mathfrak{r}'$, we can use our DLP algorithm to determine from the ideal $\mathfrak{r}_2 = (D - d^2, d)$ with distance $\delta_2 = g+1$ and the reduced principal ideal $\mathfrak{r}'$ below $(y+2)\delta_2$ the discrete logarithm $y+2$. Of these $m-1$ calls of our DLP procedure, one gives a correct answer for $y+2$ (the other $m-2$ might give a wrong or meaningless answer or no answer at all). We now have $m-1$ candidates for y and $g+1$ candidates for r. Since by (3.4) $m - 2 \le \delta(\mathfrak{a}_m, \mathfrak{a}_2) \le \delta(\mathfrak{a}_m, \mathfrak{a}_1) - 1 \le 3g$, this gives us $(m-1)(g+1) \le (3g+1)(g+1)$ candidates for $\delta(\mathfrak{r})$. We can check which one of these candidates is the correct one by using the following simple technique. If l is a candidate for $\delta(\mathfrak{r})$, compute $(Q, P, \epsilon) \leftarrow$ R-BELOW(l). If $Q = Q_r$ and $P = P_r$, compute $l + \epsilon = \delta(\mathfrak{r})$.

Assume now that we know how to compute distances and let $\mathfrak{g}$ and $\mathfrak{d}$ be reduced principal ideals such that $\mathfrak{d}$ is the reduced principal ideal below $x\delta(\mathfrak{g})$ for some $x \in \mathbb{N}_0$. Our task is to find $x \pmod{R}$. First compute $\delta(\mathfrak{g})$ and $\delta(\mathfrak{d})$. We have $0 \leq \delta(\mathfrak{d}) - x\delta(\mathfrak{g}) \leq g$, so $(\delta(\mathfrak{d}) - g)/\delta(\mathfrak{g}) \leq x \leq \delta(\mathfrak{d})/\delta(\mathfrak{g})$. If $\delta(\mathfrak{g}) > g$, then these bounds uniquely determine the integer x. If $\delta(\mathfrak{g}) \leq g$, then $\mathfrak{g} = \mathcal{O}$, in which case $\mathfrak{d} = \mathcal{O}$ and $x = 1$. □

Clearly, our three systems are broken if there is a fast algorithm for the DLP or the DP. The difficulty of the DLP was already discussed in [36], so we briefly repeat the arguments here. The most interesting situation is given by the elliptic case $g = 1$ ($\deg(D) = 4$) which also seems to be the most difficult case cryptanalytically. It was shown in [39, 40] that in this setting, there is a simple bijection from $\mathcal{R} = \{\mathfrak{r}_1 = \mathcal{O}, \mathfrak{r}_2, \ldots, \mathfrak{r}_m\}$ to the set $\{0, 2P, 3P, \ldots, mP\}$ of multiples of P (except P itself) that maps $\mathfrak{r}_1$ onto 0 and $\mathfrak{r}_i$ onto iP for $2 \leq i \leq m$. Here, P is a point on a certain elliptic curve over $\mathbb{F}_q$. Consequently, there is a polynomial-time reduction from the DLP in $\mathcal{R}$ to the DLP in the group of points on this elliptic curve: given two points P and Q on the curve with $Q = xP$ for some $x \in \mathbb{N}$, find x. If $\mathrm{char}(\mathbb{F}_q) \neq 3$, then there is also a polynomial-time reduction in the opposite direction (there may also be such a bijection or "near bijection" in the case where $\mathrm{char}(k) = 3$), so the DLP for elliptic real function fields of characteristic not equal to 3 is polynomially equivalent to the DLP for elliptic curves over a finite field. Since the best known algorithm for computing discrete logarithms on an elliptic curve over a finite field $\mathbb{F}_q$ has complexity of order $\sqrt{q}$, provided the curve is not supersingular, we require at this point exponential time to compute discrete logarithms in the set of reduced principal ideals of an elliptic function field.

For hyperelliptic real fields, there is no equivalence of the type discussed above. Here, the best known general algorithms for computing both discrete logarithms and the regulator R of the field are of complexity $O(q^{(2g-1)/5})$ (see Theorem 2.2.33, p. 78, of [39]). If $\log q \leq 2g+1$, i.e. q is again small compared to g, then discrete logarithms, including the regulator, can be computed probabilistically in subexponential time $\exp(c\sqrt{\log q^g \log\log q^g})$ where $c > 0$ is a constant ([28], Theorem 6.3.2, p. 203, of [39]). The algorithm does not appear feasible in practice; nevertheless, to be safe, one might again wish to choose q to be large relative to g. The computations in [36] show that the elliptic case $g = 1$ performed best computationally; for a 50 digit prime q, a call of R-EXP with a 50 digit exponent required 3.76 seconds on a Silicon Graphics Challenge workstation, and further optimization of this implementation will undoubtedly produce faster running times.

A Pohlig-Hellman-like technique for computing discrete logarithms in a real quadratic function field of characteristic 2 described in considerable detail in [29] can easily be adapted to work in real fields of odd characteristic. The algorithm requires knowledge of the regulator R, and as usual, its running time is essentially the square root of the largest prime factor of R. Once again, this method does not pose a threat to our cryptographic schemes at this time if q^g is sufficiently large (100 decimal digits seems more than sufficient with current computer technology) and R is not smooth.

Thus, if the parameters are chosen with some care, the fastest currently known methods for breaking our schemes are all exponential. This is in contrast to systems based on discrete logarithms in finite fields where the DLP is subexponential [2], as well as the corresponding systems in quadratic number fields (both real and imaginary), where the relevant DLPs can also be solved in subexponential time [24, 1] (assuming that the Extended Riemann Hypothesis holds). Thus, our systems might well be more secure than these other DLP-based schemes. Our key exchange protocol in real hyperelliptic fields is also significantly faster than the corresponding scheme in real quadratic number fields [12, 35] (see our computations in [36]), although we have no data available as to how our systems would perform relative to elliptic curve systems such as [5].

Unfortunately, in some instances, more information needs to be transmitted than in the original Diffie-Hellman and ElGamal systems. Let l be the size of the underlying set, i.e. $l = p - 1$ for the original Diffie-Hellman and ElGamal schemes over a finite field $\mathbb{F}_p$ and $l \approx g^g$ for our schemes. Diffie-Hellman keys require $\log l$ bits of transmission, whereas our keys are twice as long. Similarly, ElGamal signatures have size $2 \log l$, while our signatures are up to $3 \log l$ bits long. However, as mentioned before, they can be made shorter, say $2 \log l$ bits as well, if we reduce our upper bound on our parameters from $q^{g/2}$ to $q^{g/4}$. Even with these smaller quantities, we consider the schemes secure.

References

[1] C. Abel, *Ein Algorithmus zur Berechnung der Klassenzahl und des Regulators reellquadratischer Ordnungen.* Doctoral Dissertation, Universität des Saarlandes, Saarbrücken (Germany), 1994.

[2] L. M. Adleman & J. DeMarrais, A subexponential algorithm for discrete logarithms over all finite fields. *Math. Comp.* **61** (1993), 1–15.

[3] L. M. Adleman& M. D. Huang, Counting rational points on curves and Abelian varieties over finite fields. *Second Internat. Symp. Algorithmic Number Theory ANTS-II, Lect. Notes Comp. Sci.* **1122**, Springer, Berlin 1996, 1–16.

[4] L. M. Adleman, J. DeMarrais & M. D. Huang, A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. *First Internat. Symp. Algorithmic Number Theory ANTS-I, Lect. Notes Comp. Sci.* **877**, Springer, Berlin 1994, 28–40.

[5] G. B. AGNEW, R. C. MULLIN & S. A, VANSTONE, An implementation of elliptic curve cryptosystems over $\mathbb{F}_{2^{155}}$. *IEEE J. Selected Areas in Communications* **11** (1993), 804–813.

[6] E. ARTIN, Quadratische Körper im Gebiete der höheren Kongruenzen I, II. *Math. Zeitschr.* **19** (1924), 153–206.

[7] M. L. BAUER A subexponential algorithm for solving the discrete logarithm problem in the Jacobian of high genus hyperelliptic curves over arbitrary finite fields. Submitted to *J. Cryptology.*

[8] I. BIEHL, B. MEYER & C. THIEL, Cryptographic protocols based on real-quadratic A-fields (extended abstract). *Advances in Cryptology – ASIACRYPT 96, Lect. Notes Comp. Sci.* **1163**. Springer, Berlin 1996, 15–25.

[9] I. BIEHL, J. A. BUCHMANN & C. THIEL, Cryptographic protocols based on discrete logarithms in real-quadratic orders. *Proceedings of CRYPTO '94, Lect. Notes Comp. Sci.* **839**. Springer, New York 1994, 56–60.

[10] J. A, BUCHMANN, S. DÜLLMANN & H. C. WILLIAMS, On the complexity of a new key exchange system. *EUROCRYPT 90 Proceedings, Lect. Notes Comp. Sci.* **473**. Springer, Berlin 1990, 597–616.

[11] J. A. BUCHMANN & H. C. WILLIAMS, A key-exchange system based on imaginary quadratic fields. *J. Cryptology* **1** (1988), 107–118.

[12] J. A. BUCHMANN & H. C. WILLIAMS, A key-exchange system based on real quadratic fields. *CRYPTO '89 Proceedings, Lect. Notes Comp. Sci.* **435**. Springer, Berlin 1989, 335–343.

[13] D. G. CANTOR, Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.* **48** (1987), 95–101.

[14] H. COHEN, *A Course in Computational Algebraic Number Theory.* Springer, Berlin 1995.

[15] H. COHEN & H. W. LENSTRA, Heuristics on class groups. *Number Theory* (H. Jager, ed.) (Nordwijkerhout 1983), *Lect. Notes Math.* **1052**, Springer, New York 1984, 26–36.

[16] H. COHEN & H. W. LENSTRA, Heuristics on class groups of number fields. *Number Theory* (H. Jager, ed.) (Nordwijkerhout 1983), *Lect. Notes Math.* **1068**, Springer, New York 1984, 33–62.

[17] W. Diffie & M. E. Hellman, New directions in cryptography. *IEEE Trans. Inform. Theory* **22** (1976), 644–654.

[18] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **IT-31** (1985), 469–472.

[19] A. Enge, Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. Research Report CORR # 99-04, University of Waterloo, Feb. 1999.

[20] G. Frey & H. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.* **62** (1994), 865–874.

[21] C. Friesen, A special case of Cohen-Lenstra heuristics in function fields. *Centre de Recherches Mathématique Proceedings and Lecture Notes* **19** (1999), 99–105.

[22] C. Friesen, Class group frequencies of real quadratic function fields: the degree 4 case. To appear in *Math. Comp.*

[23] C. Friesen, *Bounds for frequencies of class groups of real quadratic genus 1 function fields.* Preprint.

[24] J. L. Hafner & K. S. McCurley, A rigorous subexponential algorithm for computation of class group. *J. Amer. Math. Soc.* **2** (1989), 837–849.

[25] E. Jung, *Theorie der Algebraischen Funktionen einer Veränderlichen.* Berlin 1923.

[26] N. Koblitz, Hyperelliptic cryptosystems. *J. Cryptology* **1** (1989), 139–150.

[27] A. Menezes, T. Okamoto & S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inf. Theory* **39** (1993), 1639–1646.

[28] V. Müller, A. Stein & C. Thiel, *Computing discrete logarithms in real quadratic congruence function fields of large genus. Math. Comp.* **68** (1999), 807–822.

[29] V. Müller, S. Vanstone & R. Zuccherato, Discrete Logarithm based cryptosystems in quadratic function fields of characteristic 2. *Designs, Codes and Cryptography* **14** (1998), 159–178.

[30] National Institute for Standards and Technology, *Digital Signature Standard. FIPS Publication* **186** (1993).

[31] S. Paulus & H.-G. Rück, *Real and Imaginary Quadratic Representations of Hyperelliptic Function Fields. Math. Comp.* **68** (1999), 1233–1241.

[32] S. Paulus & A. Stein, Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves. *Third Internat. Symp. Algorithmic Number Theory ANTS-III, Lect. Notes Comp. Sci.* **1423**, Springer, Berlin 1998, 576–591.

[33] S. Pohlig & M. Hellman, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Inf. Theory* **24** (1978), 918–924.

[34] R. Scheidler, Cryptography in real quadratic congruence function fields. *Proceedings of PRAGOCRYPT 1996*, CTU Publishing House, Prague (Czech Republic), 1996, 109–128.

[35] R. Scheidler, J. A. Buchmann & H. C. Williams, A key exchange protocol using real quadratic fields. *J. Cryptology* **7** (1994), 171–199.

[36] R. Scheidler, A. Stein & H. C. Williams, Key-exchange in real quadratic congruence function fields. *Designs, Codes and Cryptography* **7** (1996), 153–174.

[37] D. Shanks, The infrastructure of a real quadratic field and its applications. *Proc. 1972 Number Theory Conf.*, Boulder, Colorado, 1972, 217–224.

[38] A. Stein, *Baby step-Giant step-Verfahren in reell-quadratischen Kongruenzfunktionenkörpern mit Charakteristik ungleich 2.* Diplomarbeit, Universität des Saarlandes, Saarbrücken (Germany), 1992.

[39] A. Stein, *Algorithmen in reell-quadratischen Kongruenzfunktionenkörpern.* Doctoral Dissertation, Universität des Saarlandes, Saarbrücken (Germany), 1996.

[40] A. Stein, Equivalences between elliptic curves and real quadratic congruence function fields. *J. Théorie des Nombres de Bordeaux* **9** (1997), 75–95.

[41] H. Stichtenoth, *Algebraic Function Fields and Codes.* Springer, Berlin 1993.

[42] B. Weis & H. G. Zimmer, Artins Theorie der quadratischen Kongruenzfunktionenkörper und ihre Anwendung auf die Berechnung der Einheiten- und Klassengruppen, *Mitt. Math. Ges. Hamburg* **XII** (1991), 261–286.

[43] X. Zhang, Ambiguous classes and 2-rank of class groups of quadratic function fields. *J. China Univ. Sci. Tech.* **17** (1987), 425–431.